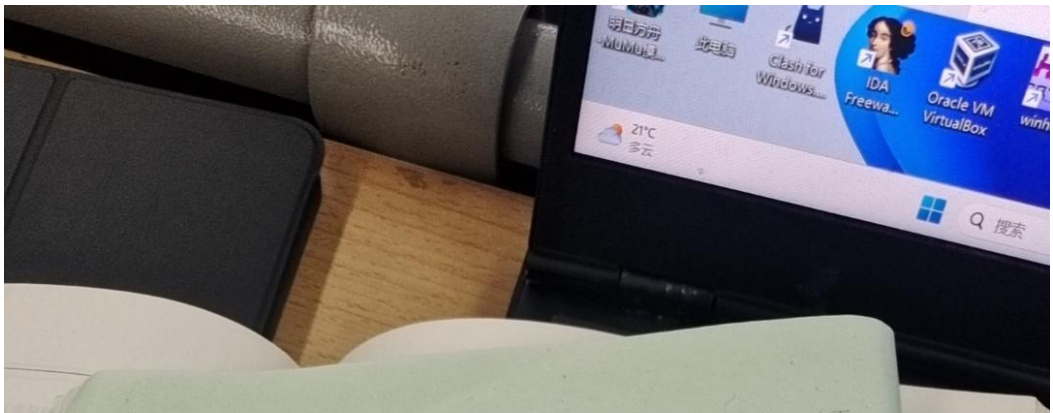

年 月 日 第 页

4.9.1 指令 相关

I_1: or r_1, r_2, r_3 RAW ~~$r_1 \to r_2/r_3$~~ $I_1 \to I_2/I_3$

I_2: or r_2, r_1, r_4 RAW r_2 $I_2 \to I_3$

I_3: or r_1, r_1, r_2 WAR r_2 $I_1 \to I_2$

WAR r_1 $I_2 \to I_3$

WAW r_1 $I_1 \to I_3$

366

4.9.2

在基本五条流水线中，WAR和WAW的依赖不会引起任何危险。在没有旁路的情况下，一条指令和下两条指令之间的任何RAW依赖

(如果寄存器读发生在时钟周期后半段，而寄存器写发生在前半段)

可以通过插入NOP指令来消除

I_1: or r_1, r_2, r_3

NOP

NOP

I_2: or r_2, r_1, r_4

NOP

NOP

I_3: or, or r_1, r_1, r_2

4.9.3 在完全旁路情况下，ALU指令可将一个值转到下一指令的EX级，而没有旁路没有NOP

4.9.4

总执行时间是时钟周期时间乘周期数 在没有任何停顿情况下，一个五条指令序列在7个周期内执行完成(第5个周期第序 接下来一周期一条)

无旁路 $(7+4) \times 180 = 1980$ps 有旁路 7×240ps $= 1680$ps 加速比 1.18

年 月 日 第 页

4.9.5 在只有ALU-ALU旁路，一条ALU指令可以转到下一条指令，但不能转到第二条指令（MEM→EX不成立）lw不能转，因为MEM决定数据，对ALU-ALU太晚所以没有NOP

4.9.6

单元旁路（7+4）×180ps=1980ps 只有ALU-ALU旁路：7×210=1470ps 加速比1.35

4.10.1

下列指令中，有一条指令在该周期内有一条加载或存储指令在使用，而不能被获取时，用XXX表示

SW r16,n(r6) IF ID EX MEM WB

LW r16,8(r0) IF ID EX MEM WB

Beq r5,r4,Label IF ID EX MEM WB

ADD r5,r1,r4 XXXXXX IF ID EX MEM WB

slt r5,r15,r4 IF ID EX MEM WB

4.10.2

只节省了一个周期，因为最后一个指令提前完成

如果有来自其他指令的lw冒险，该改变有助于消除阻塞周期

5个阶段：9个循环 4个阶段8个循环 加速比9/8=1.13

4.10.3

每个分支只带留一个周期

EX循环：4+5+1×2=11 ID循环4+3+1×1=10

加速比 1.1

0011416

年 月 日 第 页

4.10.4

5个阶段循环时间：200ps 4个阶段循环时间：210ps

加速比 9×200/8×210=1.07

4.10.5

新的ID延迟：180ps 新的EX延迟：140ps

新循环时间：200ps 旧循环时间200ps 加速比 11×200/10×200=1.1

4.10.6

EX循环：4+5+1×2=11个 新的执行时间：11/12×200ps=2200ps

MEM循环：4+5+1×3=12个 执行时间12×200ps=2400ps 加速比 0.92

4.11.1

lw r1, 0(r1) WB

lw r1, 0(r1) EX MEM WB

beq r1, r0 loop ID xxx EX MEM [WB]

lw r1, 0(r1) IF xxx ID EX MEM WB

and r1, r1, r2 IF ID xxx EX [MEM] WB

lw r1, 0(r1) IF xxx ID EX MEM

lw r1, 0(r1) IF ID xxx

beq r1, r0, loop IF xxx

4.11.2

一个阶段如果它名字在周期中不显示就是被阻塞的

每个loop循环：8，阶段循环0：0 有用：0%

年 月 日 第 页

4.12.1

对等一个next指令的依赖导致2个周期的阻塞，如果依赖于前2个next指令也是2个周期，仅依赖于第二个next指令导致一个失速周期

CPI：1+0.35×2+0.15=1.85　　0.85/1.85=46%

4.12.2

完全旁路，唯一导致阻塞的raw数据依赖是从一条指令的MEM到下一条指令的第1条next指令，只依赖导致1个失速周期

CPI：1+0.20=1.20　　0.20/1.20=17%

4.12.3

从EX/MEM旁路，EX到第1依赖可在没有阻塞下，但有任何其它依赖就会导致1个周期延迟。从EX/MEM旁路旁路，EX到2依赖不产生延迟，MEM到1仍会停泄1个周期。现在EX到1依赖要1个停泄周期因为我们必须等MEM完成才到下一条指令

EX/MEM：0.2+0.05+0.1+0.1=0.45　MEM/WB：0.05+0.2+0.1=0.35

所以MEM/WB数据阻塞更少

4.12.4

没旁路：1.85×150ps=277.5ps　有旁路：1.2×150ps=180ps

加速比：1.54

4.12.5

完全旁路：1.2×150ps=180ps　时间旅行：1×250ps=250ps　加速比：0.72

4.12.6

EX/MEM：1.45×150ps=217.5ps，MEM/WB：1.35×150=202.5ps

所以MEM/WB更小CPI

年 月 日 第 页

4.13.1

```
I1: add r5,r2,r1
    nop
    nop
I2: lw r3,4(r5)
I3: lw r2,0(r2)
    nop
I4: or r3,r5,r3
    nop
    nop
I5: sw r3,0(r5)
```

4.13.2

```
I1:
I2:
nop   向上移动插入nop
I3:
nop   这里必须多加
nop   一个nop，所以无性能提升
I4
nop
nop
I5
```

4.13.3

在有旁路时，仍需冒险检测单元，因为必须在加速指令向后续指令提供数值时插入一个周期停顿。如果没有冒险检测单元，紧随其后的加载指令的指令会得到加载指令前寄存器旧值

4.13.4

危险检测单元输出为 PCWrite、IF/IDWrite 和 ID/EXzero（在控件后输出控制 Mux 单位）~~注意~~：IF/ID 总等于 PCWrite 而 ED/EXzero 总与 PCWrite 相反。因此我们又显示每个周期 PCWrite 转发单元输出 ALUin1 和 ALUin2，控制 mux 选择第一第二输入 ALU 或是

add r5,r2,r1	IF	ID	EX	MEM	WB	PCWrite=1, ALUin1=X, ALUin2=X
lw r3,4(r5)		IF	ID	EX	MEM	PCWrite=1 ALUin1=X, ALUin2=0
lw r2,0(r2)			IF	ID	EX	PCWrite=1 ALUin1=0 ALUin2=0
or r3,r5,r3				IF	ID	PCWrite=1, ALUin1=1, ALUin2=0
sw r3,0(r5)					IF	PCWrite=1, ALUin1=0, ALUin2=0

年 月 日 第 页

4.13-5

当前处于ID阶段指令需停止，它取决于EX指令或者位于MEM阶段
所以我们需检查这两个阶段的寄存器指令。对EX，我们需检查Rd，I型指令Rd就
对MEM段，由于目的寄存器已经被选择，所以需检查寄存器号，
危险探测单元的额外输入 由寄存器来自ID/EX的管道寄存器 和输出
编号EX/MEM管道寄存器的输出寄存器 。
Rt field已经是危害检测单元的输入
不需额外输出，我们可用三个管道得出输出信号

4.13.6

add r5, r2, r1	IF ID EX MEM WB	PCWrite = 1
lw r3, 4(r5)	IF ID xxx xxx	PCWrite = 1
lw r2, 0(r2)	IF xxx xxx	PCWrite = 1
~~or r3, r5, r3~~	xxx	PCWrite = 0
or r3, r5, r3		PCWrite = 0
sw, r3, 0(r5)		

年 月 日 第 页

4.15.1

每个分支指令预测失败会导致3周期阻塞

3×(1-0.45)×0.21=0.441

4.15.2

每个分支指令预测失败导致3周期阻塞

3×(1-0.55)×0.25=0.34

4.15.3

3×(1-0.85)×0.25=0.113

4.15.4

~~3×(1-0~~ 正确预测分支的CPI为1，变成ALU指令之后CPI也是1

不正确的指令被换成为ALU指令，CPI为1

不替代：1+3×(1-0.85)×0.25=1.113 替代：1+3×(1-0.85)×0.25×0.5=1.056

加速比：~~1+33~~ 1.113/1.056=1.054

4.15.5

每个被替换的分支指令执行现在需要多一个周期

CPI不替代：1.113 替代：1+(1+3×(1-0.85))×0.25×0.5=1.181

加速比 1.113/1.181=0.94

4.15.6

正确预测：B×0.85 非循环预测正确：0.05B

~~正确~~非循环正确率：0.05B/0.20B=25%